Standardized Battery Intelligence

Robert A. Dunstan, Senior Software Engineer, Intel Mobile and Home Architecture Lab

Abstract

Intel and Duracell have jointly defined an API for smart batteries which the companies are proposing as an industry standard. Among other functions, this API enables the battery to inform the system (and user) of how much charge there is (capacity), of how long it will last (remaining operating time) and of the time to become fully charged. It also allows programmable alarm triggers. This paper will describe the specification and its proposed uses.

Introduction

Today, users of battery operated equipment regard the battery as an unpredictable source of power. On one hand, it is the only power source that allows untethered operation while on the other, it is highly unpredictable. How long will my notebook computer run? How long will it take to recharge my battery? How much longer will it run if I remove the PCMCIA cards? Have I fully charged this battery? Which of my two batteries will last longer?

Getting answers to these questions and others about the battery is a common problem for the user.

There are two broad areas where improvements to batteries will result in greater user satisfaction: 1) by giving the user a clear understanding of the battery's present capacity (e.g., how long will my computer run, how long until my battery is fully charged, etc.) and 2) by enabling effective power management systems that take advantage of accurate information supplied by the battery to extend operating time..

To address these issues, Intel and Duracell have cooperated to create the Smart Battery Data and Smart Charger specifications. In addition, Intel has created a set of ancillary software and hardware specifications. Taken together, these specifications form a common information standard for rechargeable batteries used in notebook computers, cellular telephones, camcorders and other high-end portable electronic devices. Among other benefits, this standard communication protocol opens the possibility of pack standardization and consumer replacement batteries in the future, with a resulting decrease in cost and increase in availability.

Value of a battery intelligence standard

A standard allows battery companies to focus on their core competencies: chemistries, charging algorithms, and cell characterization. A standard will also enable better power management that may result in longer run times for the user and certainly in greater user satisfaction. The Duracell-Intel standard is chemistry independent (any battery chemistry like nickel-metal-hydride or lithium ion can be safely charged) which makes introduction of new battery technologies easier. As these specifications are widely adopted, the battery will become a "standard" component, which will help make the notebook computers more attractive as competition leads to better batteries at lower cost and increased availability. From an OEM point of view, a communication standard brings design flexibility and fast access to the best battery technologies, without requiring system design changes. Furthermore, time to market for new notebook designs will be reduced, since many components (software and hardware) of the battery/charger subsystem will be reusable or available "off the shelf" from multiple vendors.

Smart Battery System Architecture

Figure 1 is a block diagram of the Smart Battery system.

Smart Battery Block Diagram

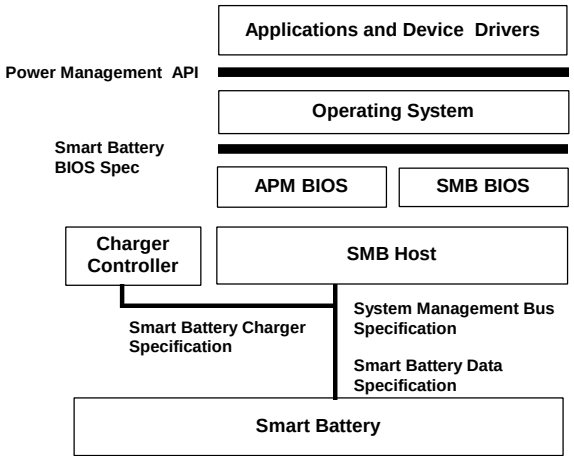


Figure 1

The SMBus

The bus used by the new Smart Battery must support multiple devices such as the host, battery, charger and other power control devices, any of which might become the bus master at any time. The reason for this requirement is that typically, the host queries the battery (e.g., host is the master) for information, but when the battery encounters a critical condition (e.g., its remaining capacity is too low), it must become the master so it can asychronously inform the host even if the host is suspended.

The battery is a very special device requiring special bus design considerations. Although the battery is capable of providing a great deal of energy, it spends much of its time in an idle state providing no energy at all. During this time, the bus should not cause any power drain. This means that the bus cannot require any power when the system is off and very little when it is suspended. Another problem, unique to the battery, is that it must be able to withstand significant ESD (electro-static discharge). The battery will not always be in the system, and will often be carried around and handled. Finally, cost (pins, complexity, gates etc.) is a critical issue because the bus will become part of a consumer product.

Existing bus architectures were studied to see if they could meet these requirements; none were found to be suitable. However, the I^2C bus (owned by Philips) met most of the requirements. With the addition of several well-defined protocols, pre-assigned device addresses and some electrical alterations, the System Management Bus (SMBus) specification was created.

Figure 2 illustrates a typical system with an SMBus.

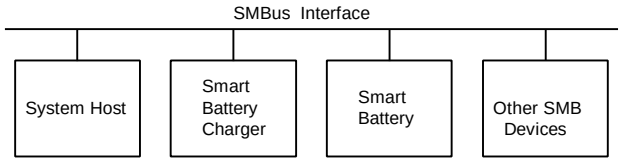


Figure 2

The Smart Battery

The Smart Battery's software API was designed to meet both the system's and user's requirements for data from the battery. The data that a smart battery must report falls into several general categories:

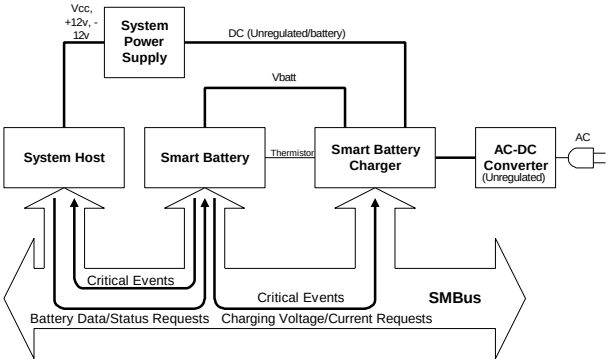
- **Alarms** - battery about to run out
- **Control/Status/Error** - control the battery's operating mode
- **Predictive functions** - how long will the battery supply power at an arbitrary rate ?
- **Measured Data** - how much current is the battery supplying ?
- **Battery State of Charge** - the fuel gauge information
- **Charging information** - how to properly charge the battery
- **Battery Characteristics** - design capacity, cell chemistry
- **Manufacturing Data** - manufacturer's name, serial number

The Smart Charger

The Smart Charger is a programmable charge controller with an API that is complementary to the Smart Battery's API. The availability of a programmable charger allows a Smart Battery to effectively control the charger thus creating a system that is truly independent of chemistry and cell construction. The charger API can be broken down into the following categories:

- **Charging** - charging voltage, charging current
- **Alarms** - process alarm messages, ignore, terminates or restart charging as appropriate

Figure 3 depicts a system powered by a Smart Battery with a Smart Charger. The host makes requests to the battery for information that is used by the SMBus BIOS, the OS or an application. The battery autonomously controls its own charging by periodically sending charging voltage and current messages to the Smart Charger. Additionally, when the battery detects an alarm or critical condition, it sends an alarm warning message to the host and the charger.



Single Smart Battery System

Figure 3

API description

Battery API

On the host side, three software entities consume the information supplied by the battery. First, the APM BIOS uses the low capacity alarm as an indication that a critical power event has occurred and supplies applications with fuel gauge information. Second, the OS, via a Smart Battery Driver, can periodically poll the SMBus BIOS for battery generated alarm messages and then process them. This driver may be part of a power management system such as the one defined in Intel's PMC (Power Management Coordinator) specification. The PMC uses information from the battery as data for its power budgeter (a software entity that allocates power based on requests from devices) and power policy managers (e.g. uses the policy to decide when to slow the CPU clock, how long a period of inactivity before spinning-down the disk, etc.) or a stand-alone driver. Finally, applications use information from the battery to present accurate run-time, charge-time and other information to the user. The battery also periodically sends charging information to the charger.

Alarms
RemainingCapacity Alarm, RemainingTimeAlarm
These are typically set and the resulting alarms are used by the APM BIOS to detect when the battery is nearly exhausted.

Control, Status and Errors
BatteryMode, BatteryStatus
The battery mode is used by the BIOS to adjust the battery's output to best match the system (e.g., report battery capacity in maH or mwH). The battery status contains a snapshot of the battery's state and last reported error.

Predictive Functions
AtRate, AtRateTimeToFull, AtRateTimeToEmpty, AtRateOK
The AtRateOK function will be used by the APM BIOS and power budgeter to see if the battery can supply enough additional energy to turn on a large power consumer (e.g., backlight, disk). The AtRate time functions are the "crystal ball" that allows an application to display information to the user about the remaining time until the battery is fully charged or how long the system will run when the power policy is changed (see figure 4).

Measured Data
Temperature, Voltage, Current, AverageCurrent
The measured data is used by special applications to calibrate the power requirements of devices for more sophisticated power management schemes such as the PMC. The temperature information may be used by a thermal budgeter or a specialized low-temperature charging regime.

Battery State of Charge
MaxError, RelativeStateOfCharge, AbsoluteStateOfCharge, RemainingCapacity, FullChargeCapacity, RunTimeToEmpty, AverageTimeToEmpty
The state-of-charge information is used by applications to give the user a fuel gauge and battery end of life indications. MaxError indicates the accuracy of the other values.

Charging Information
AverageTimeToFull, ChargingCurrent, ChargingVoltage
The host system or a charger can read the charging voltage and current. An application can display the estimated remaining charge time.

Battery Characteristics
SpecificationInfo, DesignCapacity, DesignVoltage, DeviceChemistry
This information is available for applications that display the battery's characteristics to the user or to enable special features for some types of batteries.

Manufacturing data
CycleCount, ManufactureDate, SerialNumber, ManufacturerName, DeviceName, ManufacturerAccess, ManufacturerData
This information is available for display to the user and for battery specific diagnostic programs.

Charger API

A Smart Charger is generally an SMBus slave device that only responds to messages from either the host or the battery.

Battery generated messages
ChargingCurrent, *ChargingVoltage*, *AlarmWarning*
The charging voltage and current messages allow the battery to control its own charge (default condition). The alarms are sent to both the charger and host as appropriate. The charger will discontinue charging when an alarm occurs.

Applications/Examples

User interface

The most basic application of a Smart Battery is to provide the user with an accurate fuel gauge and run time information. Figure 4 illustrates this functionality along with several important additions for the user.

A fuel gauge simply indicates how full the battery is, but fails to take into account that the capacity of a battery varies over time. A fully charged battery that is nearing its end of service life may hold only 60 to 70% as much charge as it did when new. The fuel gauge is therefore not a sufficient indicator of a battery's capacity, but rather a relative indicator for a specific battery. The Smart Battery has the ability to accurately predict how long it can supply current at any rate which allows the Power Control Panel (figure 4) to accurately display the predicted run time in addition to the fuel gauge. The user now has both the fuel gauge information AND accurate information about how long the computer will run (at the battery's present capacity and the computer's current configuration).

The Smart Battery can also accurately report how long it will take to become fully charged and this information can be displayed to the users, allowing them to decide if the battery is sufficiently charged to satisfy their needs. The final feature of the display is to illustrate how a system could accurately convert its power (current/energy) requirements for different power policies into estimated run times. The user can simply adjust the computer's power policy for economy or performance and immediately see the results by observing the change in the run time value.

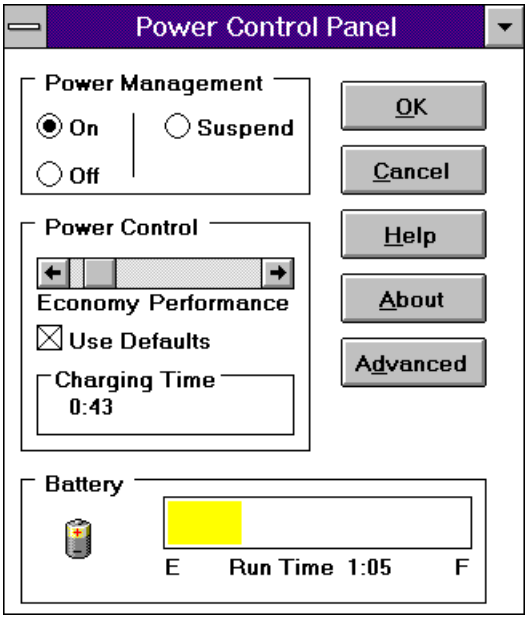


Figure 4

Figure 5 is a sample UI which could be used to display additional information about the battery powering the system.

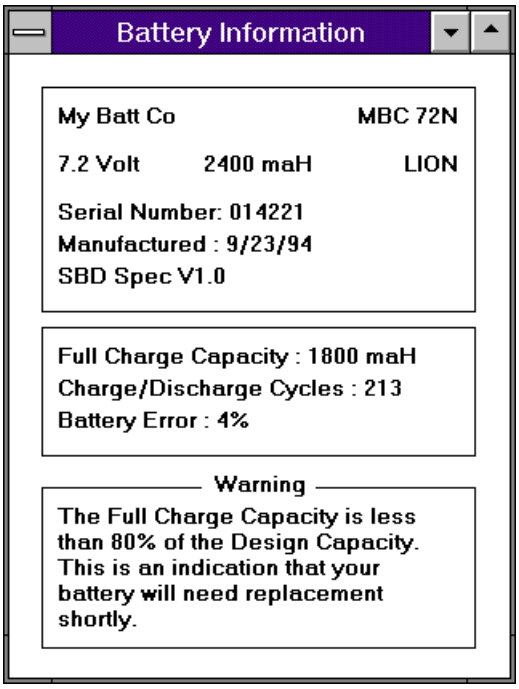


Figure 5

Alternate Charging Algorithm Example

The Smart Battery default is to send charging voltage and current messages to the charger. In a system where the temperature of the battery is an issue, the host can periodically read the battery's temperature and turn off the battery's charging messages when the battery gets above an arbitrary temperature. The host can then periodically monitor temperature and re-enable the battery's charging messages when the temperature falls to an acceptable level. In this fashion, the battery will be fully charged, but charging will take longer. This same algorithm could also be used to limit the battery's maximum temperature in critical applications such as medical devices.

Power Budgeting / Load Shedding example

A Smart Battery is capable of telling the system whether or not it can supply sufficient additional energy to service a device request such as spinning up the hard disk. This capability is particularly important when the battery is nearly exhausted. Likewise, if a large incremental load is placed on the battery, the battery's output voltage can drop low enough to cause the system's power supply to fail. This may cause the system to attempt to re-boot with the resulting loss of data. A smart battery's ability to notify the host of this potential danger can be used to prevent such accidental data loss.

OEM implementation guidelines

In order to implement the standard Smart Battery specifications in a system, several hardware pieces are needed. They are the:

- **SMBus controller**
- **Smart Battery**
- **Smart Charger**

As part of a system level architecture, Intel is also defining a set of software elements. These are the SMBus BIOS interface and a couple of OS level drivers. Taken together they will form a well-defined set of APIs at all software levels (refer to figure 6). To prove the utility of these software components as well as to provide OEMs with good software support, Intel will supply these components as part of a Smart Battery System (iSBS) development kit. This kit will include:

- **SMBus BIOS Interface TSR** - presents a work-like API
- **SMBus Emulator** - part of the above TSR
- **Parallel Port to SMBus** adapter board - used with the above to support a physical SMBus
- **Smart Battery Emulator TSR** - may be used with the above if software only emulation is required
- **Smart Battery Emulator** - runs on a separate computer using the adapter board and behaves like a Smart Battery
- **Smart Battery Demos** - Visual Basic demos
- **Device Drivers** - provide access to the Smart Battery

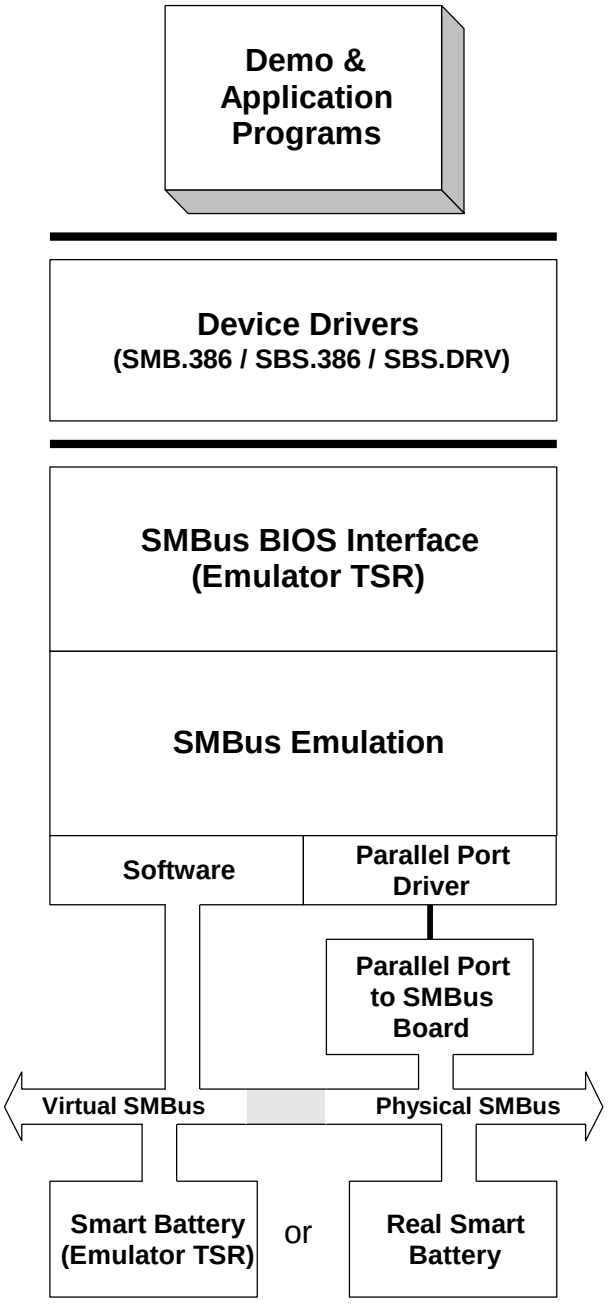


Figure 6

Discussion

The rich data available from standard smart batteries will enable more aggressive and creative power management. The end-users will benefit as better power management translates into longer run-times. They will also benefit by getting more accurate and complete information from the battery. OEMs will benefit by having the ability to create flexible designs, based on components from a variety of vendors. These designs will provide quick access to improved or new battery chemistries without requiring changes to either hardware or software.

Smart Batteries implementing the Duracell-Intel specifications will become available early in 1995. Standardizing the battery's data set and interface will significantly aid in the advancement of power management in tomorrow's notebook computers. Furthermore, accurate charging information originating from the battery allows more than one chemistry (e.g., NiMH or lithium) to be used in a given computer. This ability simply does not exist today.

Conclusion

To date, the Smart Battery standards have been well received. While the transition from today's custom solutions to a universal standard is expected to take some time, the momentum for the standard is building rapidly. Thanks to this standard, power management will become more sophisticated. OEMs will also be able to design notebooks capable of transparently accomodating advances in battery chemistries, therefore extending the life of their designs and protecting their customer's investment.

References

For further information, please refer to the following specifications available from IAL Software Operations as order number SBS5220. To order the specifications call 800-253-3696 or 503-797-4216 or email IAL_PRODUCT@ccm.hf.intel.com).

[1] Smart Battery Data Specification Revision v0.95, Duracell Inc./Intel Corporation, April 1994

[2] System Management Bus Specification Revision v0.95, Intel Corporation, April 1994

[3] Smart Charger Specification Revision v0.95, Duracell Inc./Intel Corporation, August 1994

[4] System Management Bus BIOS Interface Specification Revision v0.8a, Intel Corporation, August 1994

[5] Advanced Power Management BIOS Interface Specification Revision 1.1, Intel / Microsoft, September 1993

[6] Power Management Coordinator v1.00, Intel Corporation, April 1994